第5章

函　数

学习目标

- 掌握函数的调用过程。
- 掌握函数参数传递的方法。
- 理解局部变量和全局变量的区别。
- 掌握 datetime 模块的用法。

扫一扫

视频讲解

5.1　函数调用

5.1.1　函数的含义

函数由一段代码构成，通常这段代码具有一定的功能，且可以被另一段程序调用。例如，排序函数，函数的功能可以将几个数以从小到大或从大到小的顺序排列。在一个工程里，可以包含若干程序模块，每个模块包含一个或多个函数，每个函数实现一个特定的功能。主函数调用其他函数，其他函数之间也可以互相调用。同一个函数可以被一个或多个函数调用任意次。函数的使用能提高程序应用的模块性和代码的复用性。函数的定义形式如下。

```
def <函数名>(<参数列表>):
        <函数体>
        return <返回值列表>
```

具体说明如下。

(1) 保留字 def 用于定义函数；函数名要符合标识符的命名规则；小括号中包含的是形式参数，多个参数用逗号隔开，参数可以为空；第一行以冒号结尾。

(2) 函数体相对于 def 要有缩进。

(3) return 语句将返回值带回调用函数。

5.1.2　函数的调用过程

1. 形参与实参

函数名后面的小括号中，一般包含参数，参数可以为空。当一个函数调用另一个函数

时，调用函数中的参数称为实参，被调用函数中的参数称为形参。实参：调用函数的函数名后面小括号中的参数。形参：定义函数时，函数名后面的小括号中的参数。具体示例如下。

```
def area(width, height): # width, height 为形参
    z = width * height
    return z
w, h = 6,5
print("面积:",area(w, h))# w, h 为实参
```

2. 函数调用

从用户使用角度看，函数可以分为标准函数和用户自定义函数。从函数的形式划分，函数可分为无参函数和有参函数。

(1) 无参函数的调用。

有的函数仅实现一定的功能，如输出字符串，这时无须指定参数。

【例 5.1】 利用函数调用实现输出字符串 Hello,world!。

```
def myfun( ) :
    print("Hello,world!")
myfun( )
#运行结果
Hello,world!
```

(2) 有参函数的调用。

有参函数的调用将在 5.1.3 节中详细介绍，这里通过例 5.2～例 5.4 简单说明函数调用的过程。

【例 5.2】 使用函数调用实现求矩形的面积。

```
1. def area(width, height):            #函数定义
2.     z = width * height
3.     return z
4. w,h = 6,5
5. print("面积:",area(w, h))           #函数调用
```

在例 5.2 中，第 1～3 行，定义了函数 area(width, height)，包含两个参数宽和高(width,height)，函数功能为求矩形的面积，return 语句将 z 值返回赋给 area(w, h)，即 area(w, h)的函数值等于 z 值。第 4 行中的 w 和 h 代表矩形的宽和高，分别被赋值 6 和 5。第 5 行在输出矩形面积时，发生了函数调用，实参为 w 和 h。

① 在 area(w,h)调用函数时，实参 w 和 h 的值分别传送给形参 width 和 height，如图 5.1 所示。

② 变量 z 的值为矩形的面积，return z 是将 z 的值作为函数返回值赋值给调用它的函数 area(w,h)，即 z 的值就是 area(w,h)的函数值，如图 5.2 所示。

(3) return 语句。

return 带回返回值。无表达式的 return 相当于返回 None，函数只是完成一定的功能，没有返回值。这时，return 可以省略不写。

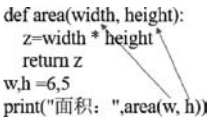

图 5.1 值传递

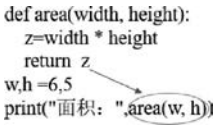

图 5.2 返回值

【例 5.3】 利用函数调用实现输出字符串。

```
def printstr( str ):
    print (str)
    return
printstr("第一次调用")
printstr("第二次调用")

#运行结果
第一次调用
第二次调用
```

【例 5.4】 利用函数调用实现求两个数的和。

```
def sum( a, b):
    total = a + b
    print ("函数内 : ", total)
    return total

total = sum( 10, 20 )
print ("函数外 : ", total)
#运行结果
函数内 : 30
函数外 : 30
```

语句 total=sum(10,20)调用 sum()函数时,实参 10 和 20 分别赋值给形参 a 和 b。在被调用函数内输出求和结果为 30,语句 return total 将变量 total 的值返回并赋给 sum(10,20),所以在函数外的输出结果也是 30。

扫一扫

视频讲解

5.1.3 参数传递

在 Python 中,变量没有类型,变量仅是一个对象的引用。例如,a 可以指向 List 类型的对象,也可以是指向 String 类型的对象,关于对象的概念将在第 6 章中详细介绍。

```
a = [1,2,3]
a = "hello"
```

1. 不可变类型参数

字符串、元组和数字类型均属于不可变数据类型,不可变数据类型的特点是一旦创建,其中的元素不能再改变,因为改变值时,会重新分配内存空间,例如,a=3,当 a 的值变

成 5 时，则重新为 5 分配内存空间。

【例 5.5】 整数类型的参数传递。

整数类型属于数字类型，例如，存在一个 int 类型的对象 2 和指向它的变量 b，在函数调用时，实参 b 的地址传给形参 a，a 也指向 int 型对象 2。在定义的函数中，执行语句 a＝10，则系统为 10 分配新的内存空间，并让 a 指向它，类似把贴在 2 上的小便签 a 撕下来贴到 10 上，变量赋值过程如图 5.3 所示。用函数 id()查看变量地址，不难看出 a＝10 前后的两条 print()输出的地址是不同的。

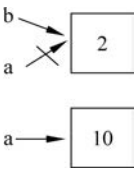

图 5.3 变量赋值过程

2. 可变类型参数

参数为可变数据类型，如 list、集合、字典等。例 5.6 以列表作函数参数实现两个变量之间的交换。

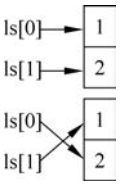

图 5.4 变量交换

【例 5.6】 列表内两个变量的交换。

本例中实参与形参同名，调用函数为 swap()，ls[0]和 ls[1]的值分别为 1 和 2。在发生函数调用时，swap()函数中 ls[0]与 ls[1]中的地址进行了交换，类似交换了小便签，因此 ls[0]和 ls[1]的值分别为 2 和 1，如图 5.4 所示。

```
def swap(ls):
    ls[1],ls[0] = ls[0],ls[1]
ls = [1,2]
swap(ls)
print(ls)
#运行结果
[2, 1]
```

3. 一致性

函数调用时，参数必须和声明时对应一致。例如，以下代码中发生函数调用时，函数 printstr()中无参数，而被调用函数中含有一个参数 str1，实参与形参不一致，运行程序时会报错。

```
def printstr(str1):
    print(str1)
    return

printstr( )
Traceback (most recent call last):

File "< ipython - input - 3 - 28eed49397f8 >", line 5, in < module >
    printstr( )
TypeError: printstr( ) missing 1 required positional argument: 'str1'
```

5.1.4 常见的函数参数

Python 语言中，函数的参数类型除了可变类型和不可变类型的参数外，常用的还有关键字参数、默认参数和不定长参数。

1. 关键字参数

关键字参数是用名称指定的参数。当参数较多时，使用关键字参数有利于标记各个参数的作用，实参顺序不必与形参顺序完全一致。在例 5.7 的示例二中，函数 print_info(age=20, name="John")中包含实参 age 和 name，与定义函数时的参数 name，age 顺序不同，但由于使用了名称指定参数，所以输出的是与实参对应的值。

【例 5.7】 关键字参数传递。

```
#示例一
def printstr( str ):
    print (str)
    return
printstr( str = "关键字参数")
#运行结果
关键字参数
#示例二
def print_info( name, age ):
    print ("名字: ", name)
    print ("年龄: ", age)
    return
print_info( age = 20, name = "John" )
#运行结果
名字: John
年龄: 20
```

2. 默认参数

调用函数时，如果没有传递参数，则会使用默认参数。在例 5.8 中，第二次发生函数调用时，print_info(name="John") 函数中只有一个实参，而函数 print_info(name,age=25)中有两个形参，在参数传递时，缺少的参数默认使用形参的参数 age，值为 25。

【例 5.8】 输出学生的年龄与姓名。

```
def print_info( name, age = 25 ):
    print ("名字: ", name)
    print ("年龄: ", age)
    return

print_info( age = 30, name = "John" )
print( ) #输出空行
print_info( name = "John" ) #age 使用默认参数
#运行结果
名字: John
```

```
年龄: 30

名字: John
年龄: 25
```

3. 不定长参数

当需要函数处理比声明中更多的参数时，需要在参数前加星号(*)，这种参数称为不定长参数，也称为可变长参数，主要有两种形式：*变量和**变量。

(1) *变量。

在函数调用时，带星号(*)的参数对应实参多余的值，实参放入一个元组中赋值给形参。如果没有指定实参，不定长参数则为一个空元组。

【例 5.9】 不定长参数传递的用法。

在下面示例一中，实参值 10 传递给形参变量 a，剩下的 20，30 形成元组(20，30)赋值给形参 p。示例二中，'Params:'赋值给形参 title，元组(10，20，30)赋值给形参 p。

```
#示例一
def star( a, *p ):
   print ("输出: ")
   print (a)
   print (p)
star( 10,20, 30 )

#运行结果
10
(20, 30)
#示例二
def print_params(title, *p):
    print(title)
    print(p)
print_params('Params:', 10, 20, 30)

#运行结果
Params:
(10, 20, 30)
```

带星号的参数如果不是放在最后，需要使用名称指定后续参数。下面代码中，实参 10 赋值给 x，调用函数 star()中的实参 z 值为 9 与形参 z 对应，实参中剩余的元素组成元组(20，30，40，50)赋值给形参 *y。

```
def star(x, *y, z):
    print(x, y, z)
star(10, 20, 30, 40, 50, z=9)
#运行结果
10 (20, 30, 40, 50) 9
```

如果单独出现星号(*)，则星号后的参数必须用关键字参数传入。在下面的代码中，

* 号后面的形参 c 需要用关键字参数传入，即实参 c 的值 3 传递给形参 c。

```
def f(a,b, * ,c):
    return a + b + c
f(1,2,c = 3)
```

(2) ** 变量。

加了两个星号（**）的参数以字典的形式传入实参。例如：

```
def star( a, **p):
   print ("输出: ")
   print (a)
   print (p)
star(1, x = 2,y = 3)

输出: #运行结果
1
{'x': 2, 'y': 3}
```

其中实参 1 赋值给形参 a，实参 x 和 y 则以字典的形式传递给有两个星号的形参 p。

(3) 混合使用。

当几种参数同时出现时，参数传递方法与(1)、(2)中分析类似，如下面代码所示。1、2、3 对应赋值给 x、y、z，同时关键字参数 a 和 b 以字典的形式赋值给形参 k。形参 p 是带星号的不定长参数，则调用函数 comb()中剩余值组成的元组(5，6，7)赋值给形参 p。

```
def comb(x, y, z, *p, **k):
        print(x, y, z)
        print(p)
        print(k)
comb(1, 2, 3, 5, 6, 7, a = 1, b = 2)

#运行结果
1 2 3
(5, 6, 7)
{'a': 1, 'b': 2}
```

在应用中，经常出现参数 * args 和 ** kwargs，这两个参数的用法与可变参数传递同理。 * args 一般放在 ** kwargs 前面。args 为元组类型，kwargs 为字典类型。

```
#示例一
def test_args(a, *args):
    print(a)
    print(type(args))
    print(args)
test_args(8, 2, 6, 7)
#运行结果
8
```

```
<class 'tuple'>
(2, 6, 7)

#示例二
def test_kwargs(a, *args, **kwargs):
    print(a)
    print(args)
    print(kwargs)
print(type(kwargs))
    test_kwargs(8, 2, 6, 7, b=3, c=4)
#运行结果
8
(2, 6, 7)
{'b': 3, 'c': 4}
<class 'dict'>
```

扫一扫

视频讲解

5.1.5 匿名函数

Python 中使用保留字 lambda 创建匿名函数，匿名函数不使用标准的 def 定义函数，匿名函数的函数体比 def 定义函数简洁，常用 lambda 表达式的形式，格式如下。

```
lambda [arg1 [,arg2,…argn]]:表达式
```

其中 arg1，arg2，…，argn 为可选参数，冒号右边为参数的表达式。

【例 5.10】 利用 lambda 表达式求和。

在 sum(10, 20)调用函数时，10 和 20 分别传递给 lambda 后的形参 a 和 b，并把求和结果 a+b 赋值给变量 sum。

```
sum = lambda a, b: a + b
# 调用 sum 函数
print ("sum = ", sum( 10, 20 ))

#运行结果
sum = 30
```

5.1.6 递归调用

递归的含义是在函数调用过程中，直接或间接地调用自身。

【例 5.11】 用函数递归调用求 n!。

```
1. def fac(n):
2.     if n<0:
3.         print("n<0,data error")
4.     elif n == 1 or n == 0:
5.         f=1
6.     else:
7.         f=n*fac(n-1)
8.     return f
9.
```

```
10. c = fac(4)
11. print("The result is {:.2f}".format(c))
```

第 1～8 行定义函数 fac(n)，该函数的功能是求阶乘。第 10 行，实参为 4，通过函数调用，参数传递给形参 n，4＞0 则执行第 7 行，函数递归调用自身，如图 5.5 所示。返回值 f，即第 10 行 fac(4)函数返回值。

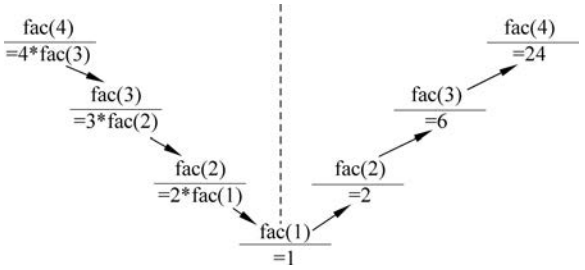

图 5.5 求 n!

扫一扫

视频讲解

5.2 局部变量与全局变量

局部变量为在函数内部定义的变量，只在函数内部起作用。如形参、实参以及函数内定义的仅在函数内使用的变量都属于局部变量。全局变量是在函数外部定义的变量，作用范围是从定义点到文件结束。

【例 5.12】 局部变量与全局变量的使用。

```
#示例一
n = 1           #n 是全局变量
def func(a, b):
    c = a * b   #c 是局部变量,a 和 b 作为函数参数也是局部变量
    return c
s = func("knock～", 2)
print(c)
Traceback (most recent call last):
File "< ipython - input - 51 - 1dd5973cae19 >", line 1, in < module >
    print(c)
NameError: name 'c' is not defined
```

因为变量 c 是局部变量，只在 func()函数内起作用。程序在函数外部输出，所以出错。如果希望让 func()函数将 n 当作全局变量，需要在变量 n 使用前显式声明该变量为全局变量，见示例二。

```
#示例二
#n = 1          #n 是全局变量
def func(a, b):
    global n
    n = b       #将局部变量 b 赋值给全局变量 n
    return a * b
```

```
s = func("knock～", 2)
print(s, n)
knock～knock～ 2
```

如果将 global n 这一行去掉，而将 n 在函数外部赋值，令 n＝1 再运行，分析运行结果。

5.3 多文件函数调用

在一个工程中常包含多个.py 文件，不同文件中的函数可以互相调用。下面举例说明同一文件夹下的两个 py 文件间的函数调用。不同文件夹下的 py 文件也可以进行函数调用。

【例 5.13】 求矩形的面积，要求 file2 调用 file1 中的面积函数。

```
#file1.py
def area(x,y):
    print('面积为:',(x*y))

#file2.py
import file1
file1.area(3,4)

#运行结果
面积为: 12
```

新建一个工程 MultiFiles，工程下建两个源文件 file1.py 和 file2.py。在 file1 中定义矩形面积函数 area(x,y)，在 file2 中用 import 导入 file1。然后，在 file2.py 中调用 file1.py，实参为 3,4，对应的 x＝3，y＝4，执行 area(x,y)函数，输出结果。file2.py 也可以写成：

```
from file1 import area
area(3,4)
```

扫一扫

视频讲解

5.4 math 与 random 库

5.4.1 math 库

math 库中包含的数学函数不适用于复数，如果需要计算复数，使用 cmath 模块中的同名函数。除特别说明外，所有函数返回值均为浮点数。内置常用运算函数如表 5.1 所示，详细的 math 库函数见附录 B。

加载 math 库，使用语句为 import math；导入某一个具体的函数，如导入开平方函数，使用语句为 from math import sqrt。这两种用法略有不同。

```
import math
math.sqrt(25)
```

```
from math import sqrt
sqrt(25)
```

表 5.1 math 库常用运算函数

函 数	数学表示	描 述
math.fabs(x)	$\|x\|$	返回 x 的绝对值
math.sqrt(x)	$\sqrt{x}$	返回 x 的平方根
math.factorial(x)	x!	返回 x 的阶乘，如果 x 是小数或负数，返回 ValueError
math.ceil(x)	$\lceil x \rceil$	向上取整，返回不小于 x 的最小整数
math.floor(x)	$\lfloor x \rfloor$	向下取整，返回不大于 x 的最大整数
math.pow(x,y)	x^y	返回 x 的 y 次幂
math.exp(x)	e^x	返回 e 的 x 次幂

【例 5.14】 利用 math 库函数计算，分步执行下列代码并查看结果。

```
import math as m
m.pi
m.exp(2)
m.pow(2,3)
m.sqrt(100)
m.fabs(-9.18)
m.factorial(5)
m.floor(3.6)
m.ceil(3.6)

#运行结果
3.141592653589793
7.38905609893065
8.0
10.0
9.18
120
3
4
```

5.4.2 random 库

Python 内置的 random 库主要用于产生各种分布的伪随机数序列。random 库采用梅森旋转算法(Mersenne twister)生成伪随机数序列，可用于除随机性要求更高的加解密算法外的大多数工程应用中。

一般函数都是基于 random.random()函数扩展而来的。使用表 5.2 中的函数时，需要导入 random 库，可用语句 from random import * 导入。

表 5.2 常用的 random 库函数

函 数	描 述
seed(a=None)	初始化给定的随机数种子，默认为当前系统时间
random()	生成一个[0.0,1.0)的随机小数

续表

函　　数	描　　述
randint(a,b)	生成一个[a,b]的整数
randrange(m,n[,k])	生成一个[m,n)以 k 为步长的随机整数
getrandbits(k)	生成一个 k 比特长的随机整数
uniform(a,b)	生成一个[a,b]的随机小数
choice(seq)序列相关	从序列中随机选择一个元素
shuffle(seq)序列相关	将序列 seq 中元素随机排列,返回打乱后的序列

【例 5.15】 生成随机数。

```
from random import *          #导入 random 库
random( )                     #生成随机数
Out[1]: 0.983090336696193
uniform(1,10)
Out[2]: 9.088254313590586
randrange(0,50,2)
Out[3]: 46
choice(range(100))
Out[4]: 68
ls = list(range(10))
shuffle(ls)
print(ls)
Out[5]: [5, 8, 6, 7, 9, 4, 0, 1, 3, 2]
```

seed()函数：指定随机数种子,随机种子一般是一个整数,只要种子相同,每次生成的随机数序列也相同。

```
seed(125)        #随机种子赋值 125
"{}.{}.{}".format(randint(1,10),randint(1,10),randint(1,10))
Out[1]: '4.4.10'

seed(125)        #随机种子仍然是 125,生成随机数同上
"{}.{}.{}".format(randint(1,10),randint(1,10),randint(1,10))
Out[2]: '4.4.10'

seed( )
"{}.{}.{}".format(randint(1,10),randint(1,10),randint(1,10))
Out[3]: '7.5.10'
```

扫一扫

视频讲解

5.5 datetime 库

datetime 库是 Python 语言用于时间处理的库,解决日期格式不统一的问题。这个库主要包含三个类：time、date 与 datetime。类和对象的概念详见 6.1 节的内容,这里不做详细介绍,仅介绍 datetime 库的基本用法。

time 类主要包含一天内的小时、分、秒、毫秒信息。date 类包含年、月、日。datetime

类是前两者的结合。

5.5.1 datetime 类型转换

在处理日期数据时，通常需要将数据集中的日期字符串转换为指定格式。datetime.strptime()方法用于将日期字符串转换为 datetime 类型，常用于数据预处理中，具体用法如下。

datetime.strptime(date_string,format)

① date_string 为日期字符串。

② format 为日期字符串的日期格式。

strptime()时间格式的表示符号由%与格式字符组成，含义如表 5.3 所示。

表 5.3 常用的 datetime()库函数

函数	说 明	函数	说 明
%y	两位数的年份表示，区间为[00, 99]	%B	本地完整的月份名称
%Y	四位数的年份表示，区间为[000, 9999]	%c	本地相应的日期表示和时间表示
%m	月份 [01-12]	%j	年内的一天 [001-366]
%d	月内中的一天 [0-31]	%p	本地 A.M.或 P.M.的等价符
%H	24 小时制小时数 [0-23]	%U	一年中的星期数[00-53]，星期日为星期的开始
%I	12 小时制小时数 [01-12]	%w	星期[0-6]，星期日为星期的开始
%M	分钟数 [00-59]	%W	一年中的星期数[00-53]，星期一为星期的开始
%S	秒 [00-59]	%x	本地相应的日期表示
%a	本地简化星期名称	%X	本地相应的时间表示
%A	本地完整星期名称	%Z	当前时区的名称
%b	本地简化的月份名称	%%	%本身

下面举例说明将日期字符串转换为 datetime 类型的方法。

【例 5.16】 将日期"2020/04/01"或"2020-04-01"转换为 datetime 类型。

示例二的"2020-04-01"中，四位的年、两位的月份和日期用“-”分隔，两位的小时、分与秒用“:”分隔，如“2020-04-01 23:04:07”。

```
#示例一
import datetime as d
date1 = "2020/04/01"
format1 = "%Y/%m/%d"
t1 = d.datetime.strptime(date1,format1)
t1
Out[1]: datetime.datetime(2020, 4, 1, 0, 0)

#示例二
import datetime as d
date2 = "2020-04-01 23:04:07"
format2 = "%Y-%m-%d %H:%M:%S"
```

```
t2 = d.datetime.strptime(date2,format2)
t2
Out[2]: datetime.datetime(2020, 4, 1, 23, 4, 7)
```

5.5.2 datetime对象的属性

1. 获取时间信息

datetime对象的属性，可以用于截取给定时间中的年、月、日以及具体时间等，见表5.4。

表5.4 datetime对象的属性

属　　性	含　　义
datetime.year	年
datetime.month	月
datetime.day	日
datetime.hour	小时
datetime.minute	分
datetime.second	秒
datetime.microsecond	毫秒

【例5.17】 使用datetime.hour得到对应日期中的小时。

```
import datetime as d
date3 = "20-01-05 23:00"
format3 = "%y-%m-%d %H:%M"
t3 = d.datetime.strptime(date3,format3)
print(t3.hour)
#运行结果
23
```

【例5.18】 利用datetime()方法计算时间差。

```
t1 = d.datetime(2020,4,1,23,0)
t2 = d.datetime(2019,4,1,22,0)
print(t1-t2)
#运行结果
366 days, 1:00:00
```

datetime对象一般不做加法运算，因为没有实际意义。

2. isoweekday()方法与datetime.now()方法

(1) isoweekday()方法。

利用isoweekday()方法可以直接查看日期对象是星期几。

```
import datetime as d
```

```
newsTime = 'Sun, 23 Aug 2020 05:15:05'
GMT_FORMAT = '%a, %d %b %Y %H:%M:%S'
newsTime = d.datetime.strptime(newsTime,GMT_FORMAT)
print(newsTime.isoweekday( ))
#运行结果
7
```

(2) datetime.now()方法。

利用 datetime.now()方法可以获得当前日期和系统时间。

【例 5.19】 通过 datetime.now()方法计算系统时间差。

第一次系统时间的输出结果为 datetime.datetime(2021,8,23,23,12,23,596904)括号中对应为年、月、日、小时、分、秒、毫秒。同理,求出第二次系统时间,两次系统时间做差。

```
import datetime
dt1 = datetime.datetime.now( )
dt1
Out[1]: datetime.datetime(2021, 8, 23, 23, 12, 23, 596904)
dt2 = datetime.datetime.now( )
dt2
Out[2]: datetime.datetime(2021, 8, 23, 23, 12, 32, 977437)
print (dt2 - dt1)
#运行结果
0:00:09.380533
```

5.5.3 date 类

date 类有三个参数:年、月、日,datetime.date(year,month,day)返回为 year-month-day。例如:

```
import datetime as d
date3 = "20 - 01 - 05 23:00"
format3 = "%y- %m- %d %H:%M"
t3 = d.datetime.strptime(date3,format3)
print(t3.date( ))
#运行结果
2020 - 01 - 05
```

利用这个属性,可以对日期数据进行批量处理,例如,把一批数据中日期在同一年的挑选出来。

5.6 实例

【例 5.20】 求 C_n^m 的值。

```
1. def fac(x):
2.     t = 1
```

```
    for i in range(1,x+1):
        t=t*i
    return t

n,m=map(int,input("please input n,m:").split(","))
c=fac(n)/(fac(m)*fac(n-m))
print("The result is {:.2f}".format(c))
```

第1～5行，定义的函数fac(x)功能为求阶乘。第8行语句中发生了3次函数调用，第一次函数调用的返回值t的值是fac(n)函数值，其余两次函数调用类似。第9行，输出的c值保留2位小数。

【例5.21】 在一行中任意输入10个数，输出这10个数中的最大值。

split()函数默认用空格分隔数据。max()函数可用于求列表元素最大值。首先，输入10个数保存在列表变量n中，创建空列表list1。然后，用i遍历列表n并把取出的列表元素添加到列表list1中。最后，用max()函数求出列表list1中的最大值。

```
n = input( )
list1 = []
for i in n.split( ):
    list1.append(int(i))
print(max(list1)) # max( )函数可用于求列表元素最大值
```

【例5.22】 有两个源文件file1和file3存在于不同路径，用函数调用实现求矩形面积。

```
#file1.py
def area(x,y):
    print('面积为:',(x*y))

#file3.py
import file1
import sys
sys.path.append('F:/file1.py')
file1.area(3,4)
```

用import导入模块时，Python语言是在sys.path中按顺序查找路径，因此需要将file1的路径添加到sys.path。

5.7 习题

一、单选题

1. 关于Python的全局变量和局部变量，以下选项中描述错误的是(　　)。

A. 局部变量指在函数内部使用的变量，当函数退出时，变量依然存在，下次函数调用可以继续使用

B. 使用global保留字声明简单数据类型变量后，该变量作为全局变量使用

C. 简单数据类型变量无论是否与全局变量重名，仅在函数内部创建和使用，函数退出后变量被释放

D. 全局变量指在函数之外定义的变量，一般没有缩进，在程序执行的全过程中有效

2. 以下选项中，不属于函数作用的是(　　)。

A. 复用代码　　B. 增强代码可读性

C. 降低编程复杂度　　D. 提高代码执行速度

3. 在 Python 语言中，关于函数的描述，以下选项中正确的是(　　)。

A. return 语句后可以没有返回值

B. Python 函数定义中必须有参数

C. 一个函数中只允许有一条 return 语句

D. def 和 return 是函数必须使用的保留字

4. 关于形参和实参的描述，以下选项中正确的是(　　)。

A. 程序在调用时，将形参复制给函数的实参

B. 参数列表中给出要传入函数内部的参数，这类参数称为形式参数，简称形参

C. 函数定义中参数列表里面的参数是实际参数，简称实参

D. 多个参数之间用空格隔开

5. 关于 lambda 函数，以下选项中描述错误的是(　　)。

A. lambda 不是 Python 的保留字

B. lambda 函数定义了一种特殊的函数

C. lambda 函数也称为匿名函数

D. lambda 函数有返回值

6. 导入模块的方式错误的是(　　)。

A. import math　　B. from math import *

C. import math as m　　D. import m from math

7. 以下程序中，输出结果不可能是(　　)。

```
from random import *
print(round(random(),2))
```

A. 0.57　　B. 0.14　　C. 0.97　　D. 1.33

8. Python 语言中，函数不包括(　　)。

A. 标准函数　　B. 第三库函数　　C. 内建函数　　D. 参数函数

9. Python 语言中，函数定义可以不包括(　　)。

A. 函数名　　B. 关键字 def

C. 一对圆括号　　D. 可选参数列表

10. 以下程序的输出结果是(　　)。

```
def func(num):
    num * = 2
x = 20
func(x)
```

```
print(x)
```

A. 40　　B. 出错　　C. 无输出　　D. 20

二、填空题

1. Python 语言中，用于定义函数的关键字是_____。

2. Python 标准库 math 中，用于计算平方根的函数是_____。

3. Python 内置函数_____用于返回序列中的最大元素。

4. Python 内置函数_____用于返回序列中的最小元素。

5. Python 内置函数_____用于返回数值型序列中所有元素之和。

三、编程题

1. 自定义函数，求两个数的最大值。

2. 自定义 lambda 函数，求两个数的最小值。

3. 编程实现：输出当前的系统时间。

4. 自定义函数，判断传入的字符参数是否为“回文”。

5. 自定义函数，计算 n 的阶乘（非递归）。

6. 自定义函数，计算 n 的阶乘（递归）。

7. 自定义函数，计算 1～n 的累加和。

8. 自定义函数，计算任意给定数的累加和（提示：使用不定长参数）。

9. 编写函数，判断一个数字是否为素数，是则返回字符串 YES，否则返回字符串 NO，并用数字 29 与 12 进行验证。

10. 编写程序，实现将列表 ls 中的素数删除并输出删除素数后列表的元素个数，ls = [23,45,78,87,11,67,89,13,243,56,67,311,431,111,141]。

11. 编程实现：计算两个日期之间的天数，输入日期的格式为 2020/12/1。

12. 编程实现：获取系统日期，并计算该日期是所在年份的第几天。

13. 自定义函数，实现输入一行字符，分别统计其中英文字母、空格、数字和其他字符的个数。

14. 自定义函数，求 m 和 n 的最大公约数。

15. 编程实现：以 123 为随机数种子，随机生成 10 个 1～999（含 999）的随机数，以逗号分隔并输出。

16. 用递归方式求解 1～100 的累加和。

17. 求 1!＋2!＋3!＋…＋n!，要求在文件 f1.py 中定义求阶乘的函数，在文件 f2.py 中求阶乘的累加和，求阶乘时需调用 f1.py 中的函数。

四、简答题

1. 简述普通参数、关键字参数、默认参数、不定长参数的区别。

2. 简述定义函数的规则。

3. 简述在 Python 语言中生成随机数的方法。